IN THE SPECIFICATION:

Please note the attached sheets indicating amendments to the specification.

At page 6, lines 12-18, change the paragraph to read as follows:

The method of the present invention, on the other hand, views data in a hierarchical database by using a relational database API. Java Data Base Connectivity (JDBC) is both a specification and an API. The JDBC specification details how the API is to behave for all relational databases. Thus, the JDBC specification details how [[the]] to handle result sets, and columns in result sets that contain all kinds of data (such as integers, strings, etc.). For example, the JDBC specification details the behavior of the JDBC driver for when a column contains an integer value. The JDBC specification does not detail how hierarchical data is handled, [[which]] but this handling is now the subject of [[this particular]] the presently described invention.

At page 10, lines 10-14, change the paragraph to read as follows:

4. DESCENDANT: [[An]] In object-oriented programming, a class (group) that is a more specialized form of another higher level class[[,]]; thus, a descendant table would be a more specialized form of a higher level table, such as a parent table.

At page 11, line 26, change the paragraph to read as follows:

14. INSTANTIATE: To create an instance of a class. An "instance" is an object in object-oriented programming, in relationship to the class to which it belongs. For example, an object MYLIST that [[to]] belongs to a class LIST, is an instance of the class "List".

At page 12, line 18, change the paragraph to read as follows:

18. JAVA APPLICATIONS: An application written in the Java language. The application can access data in a database via the JDBC driver.

At page 14, line 26, change the paragraph to read as follows:

38. NT SERVER: A computer running an operation system based on Microsoft's NT technology, such as Windows 2000 or Windows NT 4.0. The Java application JDBC driver [[(10)]] (18) and OLE DB data provider (19) all run on this computer.

At the bottom of page 16 line 24, through the top of page 17 line 1, change the paragraph to read as follows:

For example, if one table contains the fields EMPLOYEE-ID, LAST-NAME, FIRST-NAME, [[HIGHER-DATE]] HIRE DATE -- and another table contains the fields DEPT, EMPLOYEE-ID, SALARY, --- a relational database can match the EMPLOYEE-ID fields in the two tables to find such information as the names of all employees earning a certain salary on the departments of all employees hired after a certain date. Thus, the relational database uses matching values in two tables to relate information in one table to information in the other table.

At page 18, lines 15-17, change the paragraph to read as follows:

Referring now to the drawings and FIG. 1 in particular, a block diagram of a system that may use the method of the present invention is shown. PC clients 14, 15, and 16 execute software of various sorts. PC client 14 is executing internet browser software 10, PC client 15 is executing a Java application 11, and PC client 16 is executing a Java application 12 coupled with JDBC driver software 13. The Java application 12 and JDBC driver 13 can run on any computer using any operating system, such as Linux, Windows, Unix, etc, and still gain access to the database data through the JDBC driver on an MS Windows system. NT Server 20 contains a storage device 21, and typically executes software including JDBC Driver 18 and OLE DB data provider [[20]] 19. PC clients 15 and 16 connect to NT Server 20 via the JDBC driver 18. PC client 14 first connects to a J2EE application server 17 before connecting to NT Server 20 via JDBC Driver 18.

At page 19, line 25, change the paragraph to read as follows:

Referring now to FIG. 3, a flow chart that illustrates the process of determining the JDBC type for a given column in a result set is shown. This process is used to determine the "type" of the column to be returned by the ResultSetMetadata.getColumnType method defined in the JDBC API. The process begins with start bubble 41 followed by an inquiry as to whether or not the column contains hierarchical data (diamond 42). If the answer to this inquiry is no, then the column is handled in the normal fashion (block 44). The "normal fashion" refers to the sequence of handling as described in the JDBC specification cited in the Glossary. On the other hand, if the column does contain hierarchical data (YES), the value Java.sql.Types.OTHER is returned as the JDBC type of value in the given column (block 43). The process then exits (bubble 45).

At page 21, line 21, change the paragraph to read as follows:

On the other hand, if the answer to this query (at 66) is no, then a result set metadata object for this embedded result set object is created and populated with the metadata for the hierarchical data (block 67). In addition, the newly created result set metadata object is added to a collection maintained by the parent result set (block 69). This allows the newly created result set metadata object to be reused for additional rows within the same column in the result set. The process then exits (block 70).

At page 21, line 20, change the paragraph to read as follows:

With reference to FIG. 6, a flow chart illustrating the process of closing embedded ResultSet objects when the row cursor is moved is shown. The process begins with start bubble 71 followed by an inquiry as to whether or not the parent result set contains embedded result set objects (diamond 72). If the answer to this inquiry is no, then the result set is closed in the normal fashion (block 74). On the other hand, if the result set does contain embedded result set objects (YES), the process continues with a process step (block 73) to close each embedded result set object, thus preventing further access to the data contained in the result set object. The process then continues to close the result set in the normal fashion (block 73). The process then exits (block 75).